Test Magic Design Document

**Table of contents**

[1 Introduce 5](#_Toc407110208)

[2 Architecture 6](#_Toc407110209)

[2.1 Big picture 6](#_Toc407110210)

[2.1.1 Station 8](#_Toc407110211)

[2.1.2 Master 9](#_Toc407110212)

[2.2 Use cases 10](#_Toc407110213)

[2.2.1 Station 10](#_Toc407110214)

[2.2.2 Master 10](#_Toc407110215)

[3 Detailed design 11](#_Toc407110216)

[3.1 Station 11](#_Toc407110217)

[3.2 Master 11](#_Toc407110218)

**Figures** **list**

[Figure 1 - Network model 6](#_Toc407110203)

[Figure 2 - Master - Station communication 7](#_Toc407110204)

[Figure 3 - Station architecture 8](#_Toc407110205)

[Figure 4 - Master architecture 9](#_Toc407110206)

[Figure 5 - Station use cases 10](#_Toc407110207)

**Table list**

# Introduce

The Test Magic is an improvement of FATE (FPT Automation Test Embedded) project. Which is limitation in flexibility and end-user familiar.

FATE framework have following disadvantages:

* Test case was fixed on client when it created. When a PC that have planned got a problem and cannot join the plan anymore, we must setup another PC to replace. When a PC is idle, we cannot attach it into plan to share other PCs task.
* Do not flexible in test information management.
* Difficult in using & controlling.
* Database size is fast scaling.

Test Magic is more powerful automation test framework with the following main features:

* Use master – slaves model. One master on one test plan, the master responsible for manages registered slave (station), assign task for the slave depends on slave’s resource, and detects the slave connection problem.
* Flexible scheduler. Scheduler is managed by the master. The jobs in scheduler will be assigned to a corresponding slave, depends on slave’s resources. When a PC disconnected due to any network/hardware problem, the disconnected PC’s jobs will be assigned to another PC.
* Easy to control. With XMLRPC protocol, the master and slaves can be controlled from anywhere through web site or software.
* Easy test plan creator. User can easily create the test plan by update test plan information in XML/text file, then upload this file to the master, the master will create the test plan. User can view the test plan and edit directly by website if needed.
* Test plan information will be in-dependent. Each test plan and results are stored in one separated database, so when user want to re-run the test for any version in the pass, the test case information is same as the past.

# Architecture

## Big picture



Figure - Network model

The master service could run from any PC, and this PC will called the master. The master contains 2 main services, one is register service and another is station controller.

Register service allows stations register itself as build/run station into a specific test plan. After registered, the station will be used to build/run project in test plan. Whenever station is disconnected from register service, the master can detect the network status and have corresponding action with scheduler.

Station controller service manages test cases in test plan, manage test plan, and build the scheduler for each test plan.

These two services describe as the figure bellow.



Figure - Master - Station communication

### Station



Figure - Station architecture

**Physical layer**: station will use physical peripherals to communicate with hardware board, master.

**Middle ware**: third party library used for controls physical peripherals and manage system resources. Will be developed components used for interactive with master and execute test cases.

**Application**: provides interfaces through XMLRPC for master. Master can control all stations through these interfaces.

### Master



Figure - Master architecture

**Physical**: contains the peripherals that master used to communicate with stations.

**Middle ware**:

**SQLite** used as database system of the master. Each test plan can store in one database, or same database with others depends on its options.

**Django** mainly used as web application. Which allows end user easily manage project information, test cases, test plans, test result and check stations’ status.

**Scheduler** maintains multiple threads, each thread for each registered station on a test plan. Each thread manages the connection to station, get appropriate test case/suite from the test plan and run on station.

**Application**: provides interfaces which allows station register to specified test plan. Stations will send its information to master through these interfaces.

## Use cases

### Station

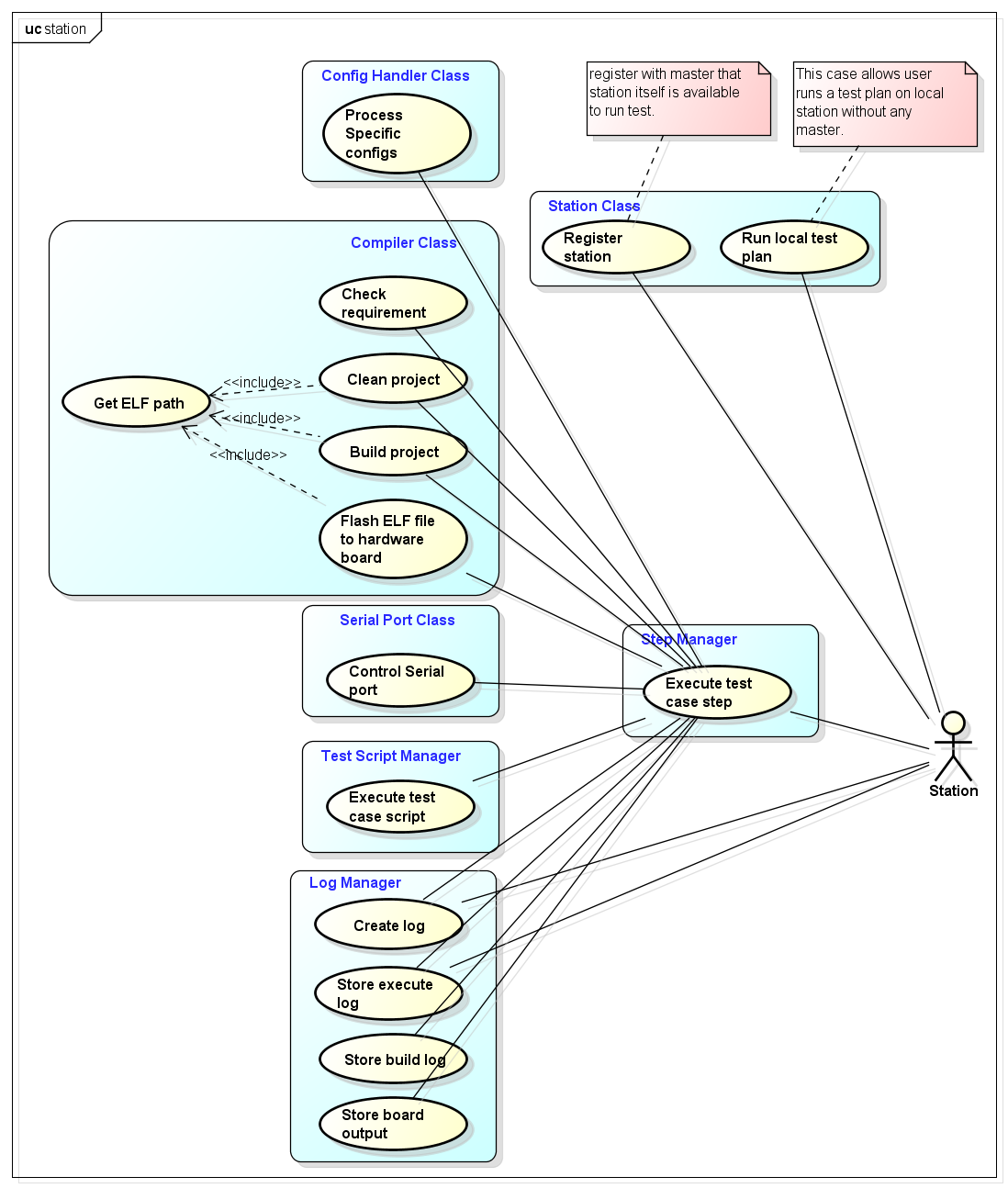


Figure - Station use cases

### Master

# Detailed design

## Station

## Master